ʕ •ᴥ• ʔ

b34r5hell

Linux and Scripting

Agenda

> Challenge environment (Linux x86_64)
> Linux CLI and Bash Shell
> Python Scripting

ʕ•ᴥ•ʔ

Challenge Environment

- Hacking involves understanding the target from the most fundamental levels in order to exploit overlooked details
- Therefore, we need to understand the environment that challenges run in
- Almost always Linux (Operating System) x86_64 (computer architecture)
- We will give an introduction to Linux and how to interact with this environment, as many challenges require you to interact with the Linux environment, either through your exploit or in order to find the exploit

ʕ•ᴥ•ʔ

Follow Along

- We will be going over the Bootcamp challenges
- We will give you time to try each section on your own
- If you have a Mac, you should be able to do all of these in “terminal”
- If on Windows and you don’t have WSL, you can run:
  - “ssh <wustl key>@shell.cec.wustl.edu”

ʕ•ᴥ•ʔ

Linux Command Line Interface (CLI)

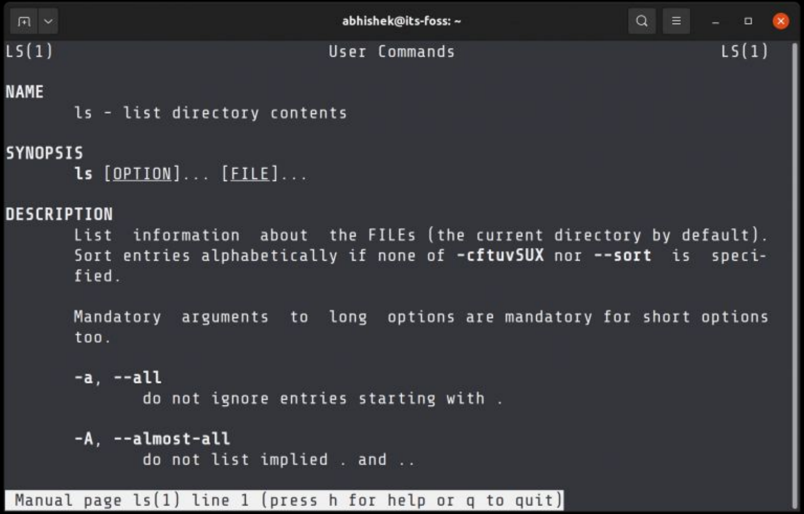
- The text-interface between user and OS
- Often use a shell as this interface (bash, zsh, dash, etc.)
- Use commands - some built-in to the shell, some installed on the system
- Gives a lot of control over the system

- Navigating the File System
  - cd
  - mkdir
  - ls
  - mv and cp

ʕ•ᴥ•ʔ

Command Flags and Man pages

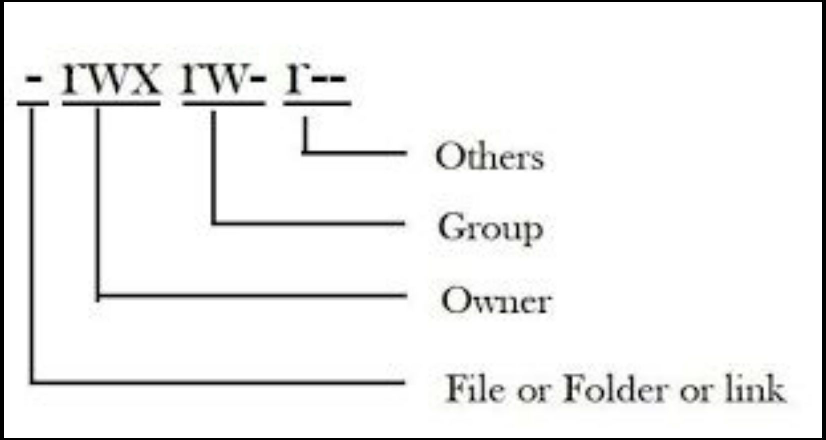
- How to figure out what a command does?
  - man pages (the manual)
  - Most Linux commands have very extensive manuals detailing their features and flags



```
LS(1)                            User Commands                           LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

 Manual page ls(1) line 1 (press h for help or q to quit)
```

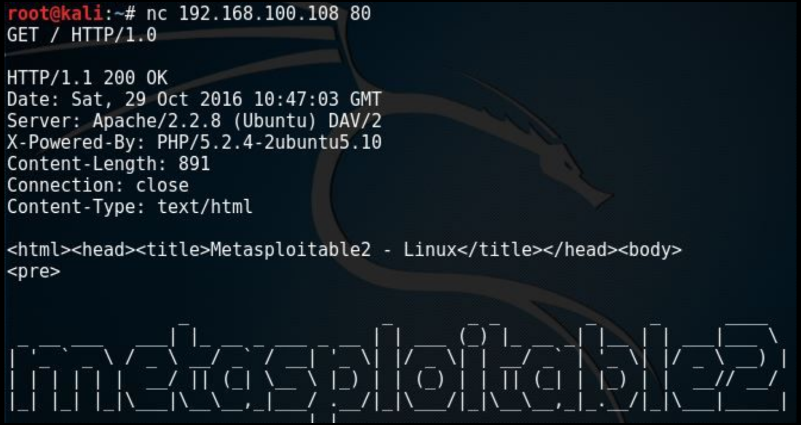
File Permissions

- File permissions are Linux's way to create security within the file system
- A file has read, write, and execute permissions for the owner, group, and other users (one bit for each = 9 total bits)
- There are also three special permissions indicated by an extra 3 bits: setuid bit, setgid bit, and sticky bit



Netcat

- Netcat
  - nc on the command line
  - Used to establish TCP (and UDP) connections to a listening server
  - Can also act as a server and listen for a client connection
  - Many challenges just give you an IP and port which can be used to communicate with the challenge server using netcat



```
root@kali:~# nc 192.168.100.108 80
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Sat, 29 Oct 2016 10:47:03 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 891
Connection: close
Content-Type: text/html

<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>

                _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
```

ʕ•ᴥ•ʔ

Piping and Redirection

- Bash (and other shells) allow for **piping and redirection**
  - Piping – “|”
    - “Pipe” the output of one command into the input of another command (only works if that command expects keyboard input)
  - Redirection – “>” or “<”
    - “Redirect” the output of one command into a file
      - Ex: echo “hey” > file
    - Can also redirect a file into the input of a command
      - Ex: cat < file

ʕ•ᴥ•ʔ

Python Scripting

- Scripting is an essential skill in CTFs
  - Many challenges are too laborious or computationally difficult to be solved by hand
  - Sometimes you need to test methodology and slowly build up your solution
  - Sometimes you make a mistake and end up having to redo your solution many more times
- We use Python because of its extensive libraries (many made just for CTFs) and its simplicity

ʕ•ᴥ•ʔ

Pwntools (https://github.com/Gallopsled/pwntools-tutorial#readme)

- Python library for exploit writing (geared towards CTFs)
- Can programmatically connect to challenge servers and send/receive data, very useful for Python scripting
- Functionality
  - Encoding/Decoding/Hashing/XOR
  - Executable reading/writing/patching
  - Assembly/Shellcode generation
  - Debugging Local Processes
  - Leaking remote memory
- Look at docs

ʕ•ᴥ•ʔ

Pwntools

- pwn.process()
  - Takes a program name and returns an object that the python script can interact with
  - Can use shell=True to run shell commands
  - readline(), sendline(), readuntil(), etc.
- pwn.remote()
  - Useful when connecting to a server, which is the case for most challenges
  - Python version of netcat

ʕ•ᴥ•ʔ

Understanding the Dojo

- We will be using pwn.college to manage challenges related to the bootcamp
- Please make an account so that you can complete the challenges
- We will go through the first few challenges so that everyone understands how the challenges are structured

ʕ•ᴥ•ʔ

Tasks

- Go through OSU’s environment setup page and setup a Linux environment
  - For most people, I recommend setting up a VM
- Setup docker on the Linux environment
- If a lot of this material was new for you, check out the Command Challenge, a great resource for practicing Linux commands
  - If you get stuck, use your resources (Google and man pages) or ask us for help
- Complete the first module in the dojo

ʕ•ᴥ•ʔ